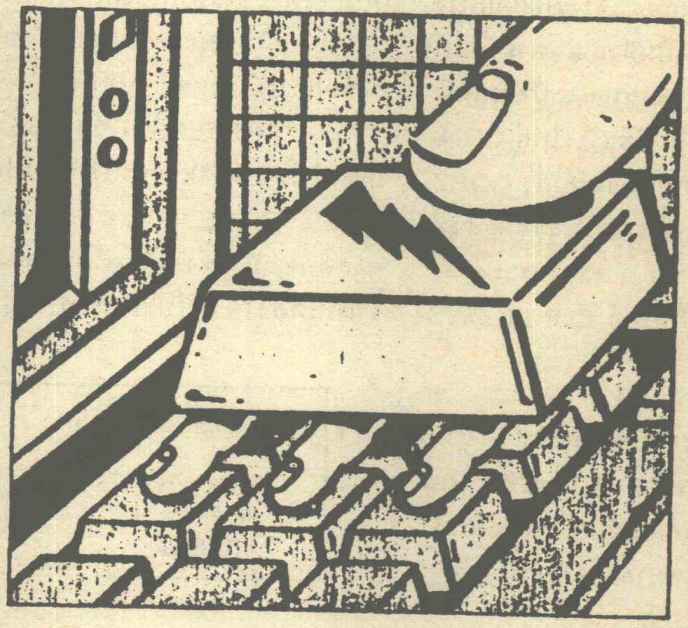
มุม OS

ตอน— CONFIG.SYS

สุรศักดิ์ สงวนพงษ์



ไฟล์ CONFIG.SYS เป็นไฟล์ที่บรรจุคำสั่งซึ่งใช้ในการกำหนดลักษณะระบบของเอ็มเอสดอส ทุกครั้งที่เปิดเครื่องใช้งาน เอ็ม–เอสดอสจะมองหาไฟล์นี้จากไดเรคเตอรีเพื่ออ่านเอาคำสั่งในไฟล์มาปฏิบัติงาน ถ้าบนแผ่นดิสค์ไม่มีไฟล์นี้อยู่ ดอสจะกำหนดลักษณะระบบตามปกติแล้วจึงทำงานอื่นต่อไป CONFIG. SYS จึงเป็นเสมือนช่องทางที่เปิดกว้างให้ผู้ใช้เปลี่ยนแปลงลักษณะระบบให้เหมาะสมกับเครื่องที่มีได้ตามต้องการ เช่น

– เปลี่ยนสภาวะการตรวจสอบการกด CTRL-C เพื่อขัดจังหวะการทำงาน

– ขยายหรือลดขนาดบัฟเฟอร์

– เปลี่ยนแปลงรูปแบบการแสดงวันเวลาตามประเทศที่ใช้

– ติดตั้งดีไวซ์ไดรเวอร์

– กำหนดจำนวนไฟล์สูงสุดที่เปิดใช้พร้อมๆ กัน

– กำหนดชื่อไดรฟ์สุดท้ายที่เรียกใช้ได้

คำสั่งใน CONFIG.SYS

ท่านผู้อ่านที่ใช้แบตช์ไฟล์มาแล้ว จะทำความเข้าใจกับ CONFIG.SYS ได้ง่าย เพราะมีแนวคิดคล้ายคลึงกัน ตัวไฟล์จะประกอบด้วยคำสั่งในดอสเอง จะมีข้อแตกต่างกันเพียงที่คำสั่งนี้จะเป็นคำสั่งเฉพาะที่ใช้คู่กับไฟล์ CONFIG.SYS เสมอ จะนำเอาคำสั่งอื่นๆ ในดอสมาใช้ไม่ได้ วิธีสร้างไฟล์ CONFIG.SYS ทำได้โดยง่าย อาจใช้คำสั่ง COPY หรือสร้างด้วยเอดิเตอร์แล้วบรรจุคำสั่งที่ต้องการลงไป ในตารางที่ 1 จะรวบรวมคำสั่งทั้งหมดที่กำหนดให้ใช้ในไฟล์ CONFIG.SYS

เปลี่ยนแปลงระบบขั้นต้น

ไฟล์ CONFIG.SYS ในรูปที่ 1 เป็นการเปลี่ยนแปลงระบบขั้นต้นอย่างง่าย คำสั่งที่อยู่ในไฟล์มีจุดมุ่งหมายการใช้งานดังนี้

– BREAK = ON ใช้เพื่อขัดจังหวะการทำงานของโปรแกรมได้ง่ายขึ้น เป็นต้นว่าโปรแกรมแปลภาษา(คอมพายเลอร์) เมื่อใช้โปรแกรมประเภทนี้เราจะหยุดการทำงานกลางคันได้ยาก เพราะโปรแกรมทำงานอยู่กับหน่วยความจำ ไม่มายุ่งเกี่ยวกับอินพุทเอาท์พุทเท่าใด (ดอสจะรับรู้การขัดจังหวะด้วยการกด CTRL-C ในขณะรอรับคีย์บอร์ดหรือกำลังส่งข้อความไปพิมพ์ที่จอภาพ)

– BUFFERS = 20 เป็นการสำรองบัฟเฟอร์ขนาด 5K (10 เท่าของปกติ)

– FILES = 20 กำหนดจำนวนไฟล์สูงสุดที่จะเปิดใช้งานได้พร้อมๆ กัน คำสั่งนี้จำเป็นต้องใช้ในโปรแกรมบางประเภทที่

1
```
A>TYPE CONFIG.SYS

BREAK=ON
BUFFERS=20
FILES=20
```

คำสั่ง	รูปแบบการใช้	จุดประสงค์
BREAK	BREAK =[ON/OFF]	ตรวจสอบการกด CTRL-C บ่อยขึ้น (ใช้ BREAK = ON)
BUFFERS	BUFFERS = X	กำหนดขนาดบัฟเฟอร์ในการอ่านเขียนดิสก์ (x มีค่าได้ตั้งแต่ 1 ถึง 99)
COUNTRY*	COUNTRY = xxx	เปลี่ยนรูปแบบการแสดงผลวันเวลาตามรหัสประเทศ
DEVICE	DEVICE = ชื่อไดรเวอร์	ติดตั้งดีไวซ์ไดรเวอร์
FCBS*	FCBS = m, n	กำหนดจำนวนไฟล์คอนโทรลบล็อกที่เปิดพร้อมกันได้
FILES	FILES = x	กำหนดจำนวนไฟล์สูงสุดที่เปิดใช้พร้อมกันได้ (x มีค่าตั้งแต่ 1 ถึง 99)
LASTDRIVE*	LASTDRIVE = x	กำหนดตัวอักษรแสดงชื่อไดรฟ์สุดท้ายที่เรียกใช้ได้
SHELL	SHELL = ชื่อไฟล์	เปลี่ยนคอมมานด์โปรเซสเซอร์

* คำสั่งที่เพิ่มเข้ามาในดอสรุ่น 3

ตารางที่ 1 สรุปคำสั่งที่ใช้ใน CONFIG.SYS

อนุญาตให้ใช้ไฟล์ได้คราวละมากๆ พร้อมกัน เช่น ดีเบสทรี

ดีไวซ์ไดรเวอร์

ดีไวซ์ไดรเวอร์ เป็นโปรแกรมที่คอยดูแลควบคุมการรับส่งข้อมูลระหว่างคอมพิวเตอร์กับอุปกรณ์รอบนอก ดีไวซ์ไดรเวอร์ทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่างเอ็มเอสดอสกับฮาร์ดแวร์ของเครื่อง เอ็มเอสดอสมีดีไวซ์ไดรเวอร์มาตรฐานเพื่ออำนวยความสะดวกให้ใช้ 2 ตัวคือ ANSI.SYS และ VDISK.SYS

การติดตั้งไดรเวอร์ทำได้โดยกำหนดไดรเวอร์เข้ากับคำสั่ง DEVICE ตัวอย่างไฟล์ CONFIG.SYS ในรูปที่ 2 เป็นการติดตั้งไดรเวอร์ ANSI.SYS

2
```
A>TYPE CONFIG.SYS

BREAK=ON
BUFFERS=20
FILES=20
DEVICE=ANSI.SYS
```

สร้างหน่วยความจำให้เป็นดิสก์ไดรฟ์

ลองจินตนาการถึงดิสก์ไดรฟ์ที่เขียนข้อมูลได้ไวยิ่งกว่าฮาร์ดดิสก์ เมื่อเวิร์ดโปรเซสเซอร์สร้างเอกสารขนาดใหญ่อาจจะใช้เวลาเพียง 3 วินาทีในการเซฟลงที่พิเศษสุดก็คือไม่ต้องเสียเงินซื้อไดรฟ์หรือแผ่นดิสก์เพิ่มเลย ทั้งนี้เพียงแต่ติดไดรเวอร์ VDISK.SYS ซึ่งเป็นไดรเวอร์จำลองดิสก์ไดรฟ์ขึ้นหนึ่งไดรฟ์ VDISK.SYS จะกันเนื้อที่หน่วยความจำส่วนหนึ่งเป็น "ดิสก์เสมือน" (virtual disk) หรือบางครั้งก็เรียกว่า "แรมดิสก์" (ramdisk) ซึ่งให้คุณสมบัติดีๆ เช่น

– อ่าน เขียนได้เร็ว เนื่องจากทำ

ามเร็วของหน่วยความจำ

– สร้างดิสค์เสมือนได้หลายไดรฟ์ เช่น มีไดรฟ์ A: และ B: อยู่แล้ว ก็สามารถสร้างไดรฟ์ C: และ D: หรือ E: เพิ่มขึ้นได้

– กำหนดขนาดของดิสค์เสมือนได้ว่าต้องการขนาดกี่กิโลไบท์ รวมทั้งสามารถกำหนดขนาดเซกเตอร์ และจำนวนไฟล์สูงสุดที่เก็บได้ในดิสค์

VICE = VDISK.SYS bbb sss ddd

พารามิเตอร์	ความหมาย	ค่าที่เป็นไปได้
bbb	ขนาดดิสค์ไดรฟ์ เป็นเลขฐานสิบที่มีหน่วยเป็นกิโลไบท์	ไม่เกินขนาดหน่วยความจำระบบ (ปกติกำหนดขนาด 64K)
sss	ขนาดเซกเตอร์เป็นไบท์	ขนาด 128, 256 หรือ 512 (ค่าปกติคือ 128)
ddd	จำนวนไฟล์สูงสุดที่ดิสค์จะเก็บได้	ตั้งแต่ 2 ถึง 512 (ค่าปกติคือ 64)

างที่ 2 พารามิเตอร์ต่างๆ ของ VDISK

บัฟเฟอร์คืออะไร

คำว่า "บัฟเฟอร์" หมายถึงหน่วยความจำส่วนหนึ่งที่สงวนไว้ใช้เป็น "กันชน" ระหว่างดิสค์กับโปรแกรม หรือเรียกว่าเป็นที่พักข้อมูล ตัวอย่างเช่น เมื่อโปรแกรมสั่งงานให้มีการอ่านข้อมูลจากแผ่นดิสค์ ข้อมูลจะถูกอ่านลงมาที่หน่วยความจำที่กันไว้เป็นบัฟเฟอร์ แล้วจึงค่อยดึงข้อมูลในบัฟเฟอร์มาใช้อีกต่อหนึ่ง ในทำนองกลับกัน เมื่อจะเขียนข้อมูลลงบนแผ่นดิสค์ ข้อมูลจะถูกส่งมาเก็บไว้ที่บัฟเฟอร์ก่อนจึงค่อยส่งไปเขียนลงบนดิสค์

ผู้ที่เคยเขียนโปรแกรมเกี่ยวกับไฟล์ข้อมูลอาจสังเกตเห็นได้ว่า เมื่อคอมพิวเตอร์ทำคำสั่งให้เขียนข้อมูลลงแผ่นดิสค์ ดูเหมือนจะไม่มีอะไรเกิดขึ้น แต่ความจริงแล้ว ข้อมูลจะถูกส่งไปเก็บที่บัฟเฟอร์ก่อน จนกว่าข้อมูลจะถูกส่งมาเต็มบัฟเฟอร์จึงค่อยลงดิสค์หนึ่งครั้ง ดังนั้นถ้ามีบัฟเฟอร์ขนาดใหญ่ก็จะประหยัดเวลาทั้งการเขียนและอ่านไปในตัว

โดยปกติดอสจะกำหนดขนาดบัฟเฟอร์ปกติไว้มีขนาด 512 ไบท์ (BUFFERS = 2) แต่ดอสก็ยอมให้ผู้ใช้เปลี่ยนแปลงขนาดบัฟเฟอร์ได้ตามต้องการ โดยสรุปแล้วบัฟเฟอร์จะมีขนาดเท่าใดขึ้นอยู่กับผู้ใช้โดยพิจารณาจากขนาดความจำที่มีในเครื่อง (ถ้าใช้บัฟเฟอร์ขนาดใหญ่จะทำให้ลดขนาดหน่วยความจำใช้งานปกติลงไป) และโปรแกรมที่ใช้งาน (โปรแกรมสำเร็จรูปบางโปรแกรมจะแนะนำว่าควรใช้บัฟเฟอร์ขนาดเท่าใด เช่น ดีเบสทรีจะแนะนำให้ใช้บัฟเฟอร์ขนาด 24)

3

```
A>TYPE CONFIG.SYS

DEVICE=VDISK.SYS 160 512 64
```

4

```
VDISK Version 1.0 virtual disk
   Buffer size:        160 KB
   Sector size:        512
   Directory entries:  64
```

ไฟล์ CONFIG.SYS ในรูปที่ 3 เป็นตัวอย่างการสร้างดิสค์เสมือนขนาด 160 K ตัวเลขต่างๆ สามารถเปลี่ยนแปลงให้เหมาะสมกับระบบที่มีอยู่ได้ โดยดูจากตารางที่ 2 ซึ่งแสดงค่าตัวเลขที่เป็นไปได้รวมทั้งค่าที่กำหนดไว้ให้ก่อน และทุกครั้งที่เปิดเครื่องใช้งานบนจอภาพจะปรากฏข้อความแสดงดังรูปที่ 4

ข้อพึงระวังในการใช้ดิสค์เสมือนนี้ก็คือ ข้อมูลในดิสค์จะอยู่ในหน่วยความจำ เมื่อปิดไฟข้อมูลจะสูญหายหมด เวลาใช้งานจึงควรมีการถ่ายเทข้อมูลกับดิสค์จริงเป็นระยะๆ

ANSI.SYS ไดรเวอร์มาตรฐาน

ANSI.SYS (American Natational Standard Institute) เป็นไดรเวอร์มาตรฐานใช้ควบคุมการทำงานของจอภาพ เช่นการเลื่อนเคอร์เซอร์ในทิศต่างๆ หรือกำหนดลักษณะพิเศษการใช้แป้นพิมพ์

ไดรเวอร์ ANSI.SYS จะสร้างฟังก์ชันมาตรฐานขึ้น ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมในภาษาระดับสูง เช่น เบสิก เพื่อเรียกใช้ฟังก์ชันมาตรฐานนี้ได้

ควบคุมการเลื่อนเคอร์เซอร์

ฟังก์ชันใน ANSI.SYS จะอยู่ในรูปของรหัสอัสกีที่ต่อเนื่องกัน และมีรหัส ESCAPE(รหัสอัสกี 27_{10}) นำหน้าเสมอ เช่น

ESC [# A เลื่อนเคอร์เซอร์ขึ้น # บรรทัด

ESC [# B เลื่อนเคอร์เซอร์ลง # บรรทัด
ESC [# C เลื่อนเคอร์เซอร์ซ้าย # คอลัมน์
ESC [# D เลื่อนเคอร์เซอร์ขวา # คอลัมน์
ESC [2J ลบจอภาพและเลื่อนเคอร์เซอร์ไปซ้ายมือบนสุด

⋮

วิธีเรียกใช้ทำได้โดยเขียนโปรแกรมส่งรหัสเหล่านี้ออกไป ในภาษาเบสิกอาจใช้คำสั่ง PRINT เข้าช่วยดังนี้

PRINT CHR$(27); "[5A" เลื่อนเคอร์เซอร์ขึ้น 5 บรรทัด

PRINT CHR$(27); "[3D" เลื่อนเคอร์เซอร์ทางซ้าย 3 คอลัมน์

PRINT CHR$(27); "[2J" ลบจอภ

ข้อสังเกต ESC มีความหมายถึง รหัสอั
27_{10} ไม่ใช่ตัวอักษร 3 ตัว และสัญลักษ
ตามหลังจะต้องอยู่ในรูปของสตริงทั้ง

มีอะไรเกิดขึ้นเมื่อกดปุ่ม

ทุกๆ ครั้งที่เราพิมพ์ตัวอักษรลงไป วงจรในแป้นพิมพ์จะสร้างตัวเลขจำนวน 2 ไบท์ขึ้นมา เพื่อส่งต่อให้ดอสตรวจสอบปุ่มกดเสมอ ANSI.SYS ก็ใช้รหัสตัวเลข 2 ไบท์นี้ มาใช้ในการสร้างคีย์บอร์ดมาโครด้วย โปรแกรมสั้นๆ ข้างล่างนี้เป็นโปรแกรมภาษาเบสิกที่แสดงถึงตัวเลข 2 ไบท์ประจำปุ่มบนแป้นพิมพ์ เช่นเมื่อกดปุ่ม F1 โปรแกรมให้ตัวเลข 0 และ 59 หรือเมื่อกด F10 จะได้ตัวเลข 0 และ 68 หรือกดตัว D จะได้เลข 44 และ 20 เป็นต้น การกดปุ่มผสมก็จะมีรหัสเฉพาะปุ่มเช่นกัน ตัวอย่างเช่นเมื่อกด ALT พร้อมกับ F1 จะได้รหัส 0 และ 104 หรือกดปุ่ม CTRL-F3 จะได้ 0 และ 96 ดังนั้น ถ้าต้องการสร้างคีย์บอร์ดมาโครให้กับปุ่มเหล่านี้ ก็เพียงแต่นำเอารหัสตัวเลขนี้ไปใช้กำหนดนั่นเอง

```
10 PRINT "PRESS ONE KEY (ESC TO STOP)"
20 DEF SEG =&H40
30 FKEY$=INKEY$ :IF FKEY$="" THEN GOTO 30
40 IF FKEY$=CHR$(27) THEN STOP
50 TAIL=PEEK(26):TAIL=TAIL-2:IF TAIL<30 THEN TAIL=60
60 CODE1=PEEK(TAIL):CODE2=PEEK(TAIL+1)
70 PRINT FKEY$;" -) ";CODE1;"  ";CODE2
80 GOTO 20

RUN
PRESS ONE KEY (ESC TO STOP)
A -)  65    30   กด A
B  )  66    48   กด B
Z -)  90    44   กด Z
; -)  0     59   กด F1
< -)  0     60   กด F2
C -)  0     67   กด F9
D -)  0     68   กด F10
^ -)  0     94   กด CTRL-F1
h -)  0     104  กด ALT-F1
Break in 40
Ok
```

เขาสร้างคีย์บอร์ดมาโครกันอย่าง

ไดรเวอร์ ANSI.SYS ยังอนุญาตใ
ใช้สร้างคีย์บอร์ดมาโครได้เอง เช่นสร้างส
DIR ด้วยการกดปุ่มเพียงปุ่มเดียว ลองดูคำ
ภาษาเบสิกง่ายๆ ดังนี้

```
PRINT CHR$(27);"[;0;68;'DIR';13P"
```

เป็นการสร้างคีย์บอร์ดมาโครให้กับปุ่ม
เมื่อกด F10 เท่ากับเป็นการพิมพ์ DIR
ด้วยการกดรีเทอร์นทันที สำหรับโปรแ
ในรูปที่ 5 เป็นตัวอย่างการสร้างคีย์บ
มาโครให้กับปุ่ม F1 ถึง F10 เป็นข้อค
ต่างๆ ผู้อ่านอาจจะลองเปลี่ยนแปลงข้อ
ต่างๆ ได้ตามต้องการ

5

```
10 FOR I=1 TO 10
20     READ CODE1$,CODE2$,M$,
30     PRINT CHR$(27);"[";COD
       :";";CODE2$;";"M$;";";
40 NEXT I
50 END
60 DATA 0,59,'DIR/W',13p
70 DATA 0,60,'CLS',13p
80 DATA 0,61,'TYPE',32p
90 DATA 0,62,'REN',32p
100 DATA 0,63,'DEL',32p
110 DATA 0,64,'CD \',32p
120 DATA 0,65,'COPY CON',32p
130 DATA 0,66,'TREE',13p
140 DATA 0,67,'CHKDSK',13p
150 DATA 0,68,'PROMPT',32p
```

ส่งท้าย

หวังว่าท่านผู้อ่านคงพอจะได้
ความคิดการใช้คำสั่งในไฟล์ CONFIG
เพื่อประยุกต์ใช้ให้เหมาะสมกับงานได้
โอกาสหน้าจะได้นำตัวอย่างการประยุ
เพิ่มเติมต่อไป